Application for Letters Patent of

the UNITED STATES OF AMERICA by –

William A. White, III
97 Sunset Road
Carlisle, MA 01741

Paul G. Wolejko
3 Willow Avenue
Newburyport, MA 01950

and

Richard Chisholm
432 Humphrey Street
Swampscott, MA 01907

Being citizens of –

THE UNITED STATES OF AMERICA

and

Luther Ullrich
Chemnitzer Street 4/B
D-61191 Rosbach, Hessen
GERMANY

Being a citizen of –

GERMANY

For:

METHOD AND APPARATUS FOR ASSIGNING A NETWORK NODE ADDRESS

Customer No.: 23569

METHOD AND APPARATUS FOR ASSIGNING A NETWORK NODE ADDRESS

The present invention is generally related to communication networks. More specifically, the present invention is related to assigning a network address to a module node in response to the location of the module within the network.

BACKGROUND OF THE INVENTION

Communication network systems are comprised of a plurality of network module nodes wherein the modules interact to accomplish a task. Proper communication within the system requires each network node, i.e., device module, to be distinguished by a unique identifier. Typically, the distinguishing identifier is a network address, e.g., Media Access Control (MAC) address, Internet Protocol (IP) address, etc.

The network identifier is manually or automatically assigned to the network module. Manual assignment requires network personnel to associate the node identifier with the network code. Automatic assignment of node identifiers can be accomplished in various ways. A technique used with BOOTP-type protocols incorporates a resource table for associating serial numbers with node identifiers. The node, i.e., client, broadcasts an address request signal containing its serial number and a server responds with a node identifier. In an Ethernet network, the node broadcasts its MAC address with an address request. An IP address is returned as the node identifier. Other similar protocols such as DHCP-type, incorporate a sequence of characters associated with the node, and an IP address will be returned in response to the address request.

Problems occur when a network node having an identifier, e.g., MAC address, is removed or replaced because the network operating code must be modified to reflect the network identifier and address of the replacing node

This invention is directed to solving these and other problems.

SUMMARY OF THE INVENTION

The present invention is directed to a method of assigning a network address to an operably connected node in response to the physical location of the node within the network. More specifically, a server detects the location of a client node and assigns a network address to the client node dependent upon the location of the node within the network.

One embodiment of the present invention is a method for assigning a network identifier to a client module node in response to the location of the client node to a network adapter, i.e., server. The method comprises powering up the network system wherein a network server assigns network addresses to one or more client nodes within the network. Initially, each client node is assigned a default network address that is a "temporary" network address. An address request is broadcasted throughout the network. The network adapter locates the nearest operably connected client node having a default address and assigns a network identifier to the node.

A further aspect of the present invention includes reserving a network identifier for later use within the network system. The reserved network identifier is associated with a holding node wherein the functionality of the holding node includes performing the addressing functions of the network.

Yet another further aspect of the present invention comprises a method for optimally assigning a network identifier to a client node.

Other advantages and aspects of the present invention will become apparent upon reading the following description of the drawings and detailed description of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

FIGURE 1 is a block diagram drawing of a communication network system of the present invention;

FIGURE 2 is an example of a program code executed within the network communication system for automatically assigning a network address to a client node during power-up of the network;

FIGURE 3 is an example of a program code executed within the network communication system for automatically assigning a network address to a client node during "hot swap" of the network;

FIGURE 4 is an example of a program code executed within the network communication system for optimizing the program shown in FIGURE 3;

FIGURE 5 is an example of a program code executed within the network communication system for updating an activity table, $ACT; and,

FIGURE 6 is a table of constructs utilized in one embodiment of the present invention, i.e., CANOpen, network.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

While this invention is susceptible of embodiments in many different forms, there is shown in the drawings and will herein be described in detail a preferred embodiment of the invention with the understanding that the present disclosure is to be considered as an exemplification of the principles of the invention and is not intended to limit the broad aspect of the invention to the embodiment illustrated.

FIGURE 1 depicts an embodiment of the present invention directed to a communication network system10, preferably CANOpen. At least one client node 12 is communicably connected to a network adaptor 14, e.g., server node, field bus connector. Each client node 12 is operably connected to the server node 14 via a communication bus 16, preferably CANOpen. Additionally, an address bus 18 connects the plurality of client nodes 12 with the server node 14. Each client node 12 is operably connected to the address bus 18 via an address input pin 20 and an address output pin 22. The address output 22 of the sever 14 is connected to the address input 20 of the nearest, adjacent, client node 12. The address output 22 of the adjacent client node 12 is connected to the address input 20 of the next adjacent client node 12. This serial-like connection continues with subsequent client nodes 12 until all of the client nodes of the network 10 are operably connected to the address bus 18. The address input 20 and address output 22 of each client node 12 resides in an active or inactive state and are operably connected to a processor 24 within the client node.

Preferably, the network 10 can be conveniently configured by inserting various client nodes 12, e.g., device modules, in a DIN rail 26. The DIN rail 26 operably connects the plurality of client nodes 12 with one or more server nodes 14. The DIN rail 26 comprises a power bus 28, the communication bus 16, and the address bus 18. Each client node 12 can be easily snapped into and out of the DIN rail 26.

The server node 14 manages the process of assigning a network identifier to each client node 12 operably connected to the network 10. The network identifier facilitates communication among the network nodes 12, 14. Similar to the client node 12, the output address pin 22 of the server node 14 is capable of having an active or inactive state. Normally, the client node's output address pin 22 is in the inactive state, however, the server node 14 can set the state to active. The state of the input 20 and output 22 address pin is responsive to the client node's processor 24 and to the output address pin 22 of the server node 14. Generally, the output address pin 22 of each client node 12 is in the inactive state. The output address pin 22 can be set active by a server node command, or when the server node 14 sets its own output address pin active. Thus, the server node

14 is aware of the state of each network node's 12 output address pin 22.

Each client node 12 utilized in the network 10 is selected in response to the functionality desired. The client node 12 can sense whether its input address pin 20 is inactive or being forced active. The client node 12 is capable of setting its output address pin 22 to active or allowing the status of the output address to fall inactive. The current address identifier of the client node 12 is stored in the node's memory 30. The memory 30 keeps a record of the amount of active/inactive transitions of the input address pin 20. At system power-up, the client nodes 12 are initialized with a default network identifier. The default network identifier is a temporary network identifier of the client node 12 and retained until a more permanent identifier, i.e., network address, can be assigned to the client node.

The server 14 may begin the assignment process upon receiving an address request from a client node 12. The server 14 can also initiate the assignment process upon a network power-up or a hot-swap of a client node 12. The address request transmitted by the client node 12 notifies any server node 14 on the network 10 that a client node has joined the network. At this time, the client node 12 is unconfigured for communication on the network 10. Essentially, after requesting the address identifier, the client node 12 is a passive participant merely responding to the messages of its server node 14 during the process of assigning the network address.

Besides assigning an address identifier to a client node 12 residing on the network 10, the server node 14 is capable of transmitting various messages to the client node. For instance, the server node 14 can also change or toggle the status of a client node's 12 output address pin 22.

Each client node 12 is capable of broadcasting an address request throughout the network 10. In response to the address identifier request, the server node 14 will assign an exclusive address identifier to replace the default identifier of each client node 12. Each client node 12 includes a memory location 30 for storing the assigned network identifier and the amount of inactive-active transitions of its input address pin 20 as toggled by the server 14. When requested, the client node 12 can provide the amount of toggles, i.e., state transitions, to the server node 14.

Initially, the client node identifier is unconfigured at power-up and each output address pin 22 is inactive. Each client node 12 contains a default value in its network identifier memory location 30; preferably the default value is 127. During power-up, the server node 14 reassigns the address of every client node 12 operably connected to the network 10. The client node 12 transmits a network broadcast message requesting a

network address identifier and identifying itself by its default value. The server node 14 transmits a toggle signal having a predetermined amount of active-inactive transitions. The toggle signal is sent to any client node 12 having a default identifier. The nearest client node 12 receiving the toggle message will transition its output address status between active and inactive the amount of times sent by the server 14. The client node 12 will also store the amount of toggles in its memory 30. The server 14 then transmits a signal to obtain the amount of state transitions of the client node 12 having a default address. If the server 14 receives an amount of state transitions from the client node 12 matching the amount of state transitions transmitted, by the server node, the server will reassign the client node's address identifier from default.

Since all client nodes 12 have a default address upon power-up, the nearest client node to the server 14, e.g., the adjacent client node, will respond to the server's transition request signal. This client node 12 will then be reassigned a network address identifier. The output address pin 22 of the newly identified client node 12 will be set inactive. The server 14 will then receive another address request signal from another client node 12 that has yet to be assigned a replacement address identifier for its default identifier. The server node 14 will transmit another toggle signal to the default address and these steps will be repeated until all client nodes 12 having a default address have been reassigned a more permanent address identifier. An example of an algorithm utilized by the server node 14 during power-up of the network 10 is shown in FIGURE 2.

Another aspect of the present invention involves assigning a network address identifier to a client node 12 actively inserted into a powered-up network 10, i.e., "hot swap." In this situation, the network has been powered up and the power-up client node assignments have been completed. One or more additional client nodes are inserted into the network. The newly added client nodes 12 will power-up upon insertion into the network 10 and broadcast an address request to the network. Because the address identifier of the added client nodes 12 is unconfigured, the default address identifier will be used in the request message transmitted by the newly added client node. FIGURE 3 depicts an algorithm utilized during assignment of network identifiers to client nodes 12 in a "hot-swap" configuration.

Yet another aspect of the present invention is directed to optimizing the assignment of an address identifier for inserted, swapped, or substituted, client nodes 12, i.e., "hot swap." FIGURE 4. The optimization process monitors the history of the network communication and utilizes the information to identify a client node 12 most likely requesting an address identifier. The server node 14 assumes that the client node

12 least recently communicated with is an optimal candidate to have been replaced with an operable device module 12, and thus more likely to be requesting an address identifier. Although the optimization process is similar to the non-optimized procedure, the optimization process assumes that the client node 12 that was least recently communicated with is the most probable candidate to be requesting an address identifier.

Steps 1-4 and 9.4.2.1 of the "hot swap" algorithm shown in FIGURE 3 are used to detect which client node is missing. Utilizing an activity table, $ACT, the server node 14 begins by checking the most probable client nodes 12 requiring an address identifier, rather than systematically stepping through each client node in sequence, with respect to physical positioning on the network 10. FIGURE 4.

The activity table is an array indexed by client node addresses. Each entry in the activity table includes several variables, i.e., $OLDER, $NEWER, $OLDEST, and $NEWEST. $OLDER is the node identifier, $NODEID, of the client node 12 that was communicated with immediately previous to the present client node. The client node 12 is unconfigured when it is the "oldest" in the activity table. NEWER is the node identifier of the client node 12 that is communicated with immediately following. This client node 12 is unconfigured immediately after a new message is received from this client node and the activity table is updated. $NEWEST and $OLDEST are two variables used to contain the node identifiers of the client nodes 12 that are most recently and least recently communicated with. For example, an array and structure notation is used to refer to activity table entries, e.g., $ACT[5]. $OLDER contains the identifier of the client node 12 that communicated with the server node 14 immediately prior to client node 5.

The activity table is initialized as follows: the $OLDER fields for client nodes 1 through $MAXNODES-1 are set to the client node identifier plus 1; the $OLDER field of the $MAXNODES client node is set to $UNCONFIGURED; the $NEWER field of the first client node is set to $UNCONFIGURED; and, the $OLDEST fields of client nodes 2 - $MAXNODES are set to the client node identifier minus 1.

When a message is exchanged with a client node 12 on the network, the server node 14 updates the activity table to reflect that the client node now has been involved in the most recent communication. Each time the server node 14 sends or receives a message, it updates the activity table. An example of an algorithm for updating the activity table is shown in FIGURE 5.

Yet another further aspect of the present invention is directed to reserving a location(s) , e.g., client node placeholder address, within the network 10 for later

population of device modules 12. This functionality is extremely useful for designing expandable networks that can be later upgraded. This ability provides a network designer the flexibility to configure the network 10 even though some of the equipment modules 12, 14 are not ready to be installed. Additionally, this ability can be utilized to compensates for a node failure wherein the node must be removed and a placeholder node can be inserted to reserve the address location until repair or replacement of the network module can be made. The placeholder comprises an input address pin and an output address pin. Both address pins are operably connected to the address bus 18 wherein the placeholder node inexpensively functions to reserve a location on the network 10 for later insertion of a module node 12.

The placeholder node includes two primary characteristics. First, the placeholder strips one transition cycle, i.e., inactive/active/inactive, off the toggle signal and must permit the remaining transitions of the toggle signal to pass through, i.e., from the placeholder node's input address pin to its output address pin. And second, the state of the placeholder node's input address pin is copied through the placeholder node to its output address pin wherein after a series of transitions, the state of the output address pin can be held active for a predetermined amount of time.

Assuming an unaddressed client node 12 is adjacent the placeholder node, the unaddressed client node will receive and store, a value of one less transition of the toggle signal transmitted by the server node. The default addressed client node will respond to the server's request for toggle transitions and will then be assigned an address identifier with a value equal to 1 less than the address identifier assigned to the placeholder node located one physical position closer to the server node 14.

Placeholder nodes may be positioned adjacent to each other. Each placeholder node will then remove one inactive/active/inactive transition of the toggle message, thus leaving an available network address for each held location.

When a client node 12 is inserted to replace a placeholder node, the client node will be assigned an address identifier through the "hot swap," preferably optimized, method described above.

While the specific embodiment has been illustrated and described, numerous modifications come to mind without significantly departing from the spirit of the invention, and the scope of protection is only limited by the scope of the accompanying claims.